# Evaluation of Crossover Operator Performance in Genetic Algorithms with Binary Representation

Stjepan Picek, Marin Golub, and Domagoj Jakobovic

Faculty of Electrical Engineering and Computing, Unska 3, Zagreb, Croatia

**Abstract.** Genetic algorithms (GAs) generate solutions to optimization problems using techniques inspired by natural evolution, like crossover, selection and mutation. In that process, crossover operator plays an important role as an analogue to reproduction in biological sense. During the last decades, a number of different crossover operators have been successfully designed. However, systematic comparison of those operators is difficult to find. This paper presents a comparison of 10 crossover operators that are used in genetic algorithms with binary representation. To achieve this, experiments are conducted on a set of 15 optimization problems. A thorough statistical analysis is performed on the results of those experiments. The results show significant statistical differences between operators and an overall good performance of uniform, single-point and reduced surrogate crossover. Additionally, our experiments have shown that orthogonal crossover operators perform much poorer on the given problem set and constraints.

## 1 Introduction

When dealing with optimization problems, one of the challenges is the need to avoid trapping of the algorithm in the local optima. That objective is especially present when there are numerous local optima and the dimensionality of the problem is high. In the last 50 years there have been many metaheuristics coming from the evolutionary computation family that are successful when dealing with such problems. Among those algorithms, genetic algorithms has received considerable attention. During that time, genetic algorithms have been successfully applied to the variety of optimization problems. Since the invention of genetic algorithms in 1960s by J. Holland, crossover operator have played major role as an exploratory force of genetic algorithms. Holland also used mutation operator in his work, but it was generally treated as subordinate to crossover operator [6]. In the following years, many modifications of crossover operators have appeared [3] [11] .

Firstly, it is necessary to answer the question of whether it is possible to find the best search algorithm. The answer is no, since the "No free lunch" theorem demonstrates that when averaged over all problems, all search algorithms perform equally. However, when working with some knowledge about the problem,

it is possible to choose more suitable algorithms.
To evaluate the performance of the algorithms it is not enough to compare mean and standard deviation values [5]. Rather, a proper statistical analysis should be performed. In this paper, we use *nonparametric* statistical tests to evaluate the performance of the crossover operators. For a justification on the selection of the employed statistical methods, and further information about the statistical methods, refer to [2] and [13] .

The significance of this comparison lies in the fact that a thorough statistical analysis of these crossover operators has not been done before, to the best of our knowledge. The tests done previously were conducted on smaller sets of crossover operators. The main disadvantage with that approach lies in the fact that it is based on the mean, and standard deviation values, or on the parametric statistical tests. The parametric statistical tests are usually performed without the checking of the necessary conditions for their use, especially in the multi-problem analysis scenario [5]. In this paper we use a mathematically appropriate and thorough approach in conducting statistical analysis. We begin this paper by giving a short overview of the relevant theory in Section 2. Section 3 defines the parameters used in the experiments, the results obtained from the experiments and gives directions for the future work; finally, Section 4 draws a conclusion.

## 2 Preliminary

### 2.1 Crossover Operator

Crossover, as a process where new individuals are created from the information contained within the parents, is often said to be the distinguishing feature of genetic algorithms. Crossover operators are usually applied probabilistically according to a crossover rate $p_c$. In this paper crossover will refer to a two-parent case i.e. two individuals are selected as the parents to produce one offspring. Table 1 lists references for all the crossover operators investigated in this paper.

### 2.2 Test Functions

Test functions from the Table 2 have been selected for the investigation of crossover efficiency. These functions represent well known problems for evaluating the performance of evolutionary algorithms. The table shows the function formulae, dimension of the problem as used in the experiments, optimum value of the function and the reference where additional information on the function can be found.

### 2.3 Evolutionary Computation Framework - ECF

Evolutionary Computation Framework (ECF), used in this work, is a C++ framework intended for the application of any type of the evolutionary computation. ECF is developed at the Faculty of Electrical Engineering and Computing, Zagreb [8].

**Table 1.** Crossover operators used in experiments

| Crossover operator | Reference |
|---|---|
| Single-point crossover | [11] [16] |
| Two-point crossover | [16] |
| Half-uniform crossover | [4] |
| Uniform crossover | [11] [16] |
| Shuffle crossover | [3] |
| Segmented crossover | [3] [10] |
| Reduced surrogate crossover | [3] |
| Non-geometric crossover | [7] |
| Orthogonal array crossover | [1] [9] [17] |
| Hybrid Taguchi crossover | [15] |

**Table 2.** Benchmark functions ($D = 30$)

| Test function | Domain Range | Reference |
|---|---|---|
| $f(x) = \sum_{i=1}^{D} x_i^2$ | [-5.12, 5.12] | [12] |
| $f(x) = \sum_{i=1}^{D} i \cdot x_i^2$ | [-5.12, 5.12] | [12] |
| $f(x) = \sum_{i=1}^{D} 5 \cdot i \cdot x_i^2$ | [-5.12, 5.12] | [12] |
| $f(x) = \sum_{i=1}^{D} \left( \sum_{j=1}^{i} x_j^2 \right)$ | [-65.536, 65.536] | [12] |
| $f(x) = \sum_{i=1}^{D-1} 100 \cdot \left( x_{i+1} - x_i^2 \right)^2 + (1 - x_i)^2$ | [-2.048, 2.048] | [1] |
| $f(x) = 10 \cdot D + \sum_{i=1}^{D} \left( x_i^2 - 10 \cdot \cos(2 \cdot \mathit{\Pi} \cdot x_i) \right)$ | [-5.12, 5.12] | [12] |
| $f(x) = \sum_{i=1}^{D} -x_i \cdot \sin\left( \sqrt{\lvert x_i \rvert} \right)$ | [-500, 500] | [9] |
| $f(x) = \sum_{i=1}^{D} x_i^2/4000 - \prod_{i=1}^{D} \cos\left( x_i/\sqrt{i} \right) + 1$ | [-600, 600] | [1] |
| $f(x) = -20 \cdot e^{-0.2\sqrt{\sum_{i=1}^{D} x_i^2/D}} -$<br>$-e^{\sum_{i=1}^{D} \cos(2\mathit{\Pi} x_i)/D} + 20 + e$ | [-32.768, 32.768] | [12] |
| $f(x) = -\sum_{i=1}^{D} \sin(x_i) \cdot \left( \sin\left( i \cdot x_i^2/\mathit{\Pi} \right) \right)^{20}$ | [0, 3.14] | [12] |
| $f(x) = \sum_{i=1}^{D} \left( 10^6 \right)^{(i-1/D-1)} \cdot x_i^2 - 450$ | [-100, 100] | [14] |
| $f(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$ | [-10, 10] | [9] |
| $f(x) = \sum_{i=1}^{D} 2 \cdot D +$<br>$\sum_{i=1}^{D-1} \left[ \sin(x_i + x_{i+1}) + \sin(2 \cdot x_i \cdot x_{i+1}/3) \right]$ | [3, 13] | [1] |
| $f(x) = 1/D \cdot \sum_{i=1}^{D} \left( x_i^4 - 16 \cdot x_i^2 + 5 \cdot x_i \right)$ | [-5, 5] | [9] |
| $\sum_{i=1}^{D} \left( \sum_{k=0}^{20} \left[ 0.5^k \cdot \cos\left( 2\mathit{\Pi} \cdot 3^k (x_i + 0.5) \right) \right] \right) -$<br>$-D \sum_{k=0}^{20} \left[ 0.5^k \cdot \cos\left( \mathit{\Pi} \cdot 3^k \right) \right]$ | [-0.5, 0.5] | [14] |

## 3 Experimental Results and Comparisons

### 3.1 Environmental Settings

In all the experiments, binary-coded genetic algorithm with roulette-wheel selection [16] is used. Individuals are binary vectors which represent real values [10]. Parameters of the genetic algorithm that are common to every round of the experiments are as following: simple bit mutation with mutation probability $p_m$ of 0.01 per bit, population size $N$ of 30, precision is set to 3 digits after the decimal point, which is sufficient to produce large enough number of possible solutions for used test problems. The number of independent runs for each experiment is 30, dimensionality $D$ of all the test problems is set to 30, and the number of fitness evaluation is set to 500000. For all the test functions finding the global minimum is the objective. As a performance measure, we use the error rate obtained for every algorithm.

Two rounds of the experiments are conducted in total. Each round is designed to provide the answer to one question. *The first round* is a parameter tuning round where the goal is to find the best value of $p_c$ for every crossover operator and every test function. The goal of *the second round* is to find the best overall algorithm on the set of all the test functions used in this paper.

Additionally, one may want to find the best performing operator for each of the test functions. The answers to these questions are not included in the paper, because knowing the 'best' operator for a single problem may be relevant only for that same problem and the volume of those results exceeds the scope of this work. Naturally, by choosing other selection and mutation operators it is possible to expect different results of analysis. However, in order to adhere to prescribed paper length we decided to use the simplest mutation operator with constant $p_m$ value since the mutation operator is not of the primary interest in this paper. Furthermore, we use only roulette wheel selection as an example of commonly used selection type where the $p_c$ are variable.

### 3.2 Experiments and Results

In the first round, each operator is run 30 times on each test function with different values of $p_c$, ranging from 0.1 to 1 in steps of 0.1. The results for every combination of single operator and test function are then processed to find the best crossover probability value. In a large number of combinations there were no significant statistical differences between performances for different $p_c$ values. However, since a single probability value is needed if there is to be an operator comparison, for every combination the $p_c$ value which gives the smallest average error on best individuals in 30 runs was chosen.

In the second round, the operators were compared using the above average error of best individuals from 30 runs with the best probability value. This structure of input data is in accordance with previous analysis performed over multiple algorithms and test problems [5] [14]. A series of non-parametrical statistical tests is performed on the data in this round.

The first test is Friedman two-way analysis of variances by ranks, which represents the most well known procedure for testing the differences between more than two related samples [13]. Additionally, we use Iman-Davenport test as a variant of Friedman test that provides better statistics [2]. The objective of the Friedman and Iman-Davenport tests is to show that there is a statistical difference between groups (crossover operators). If there is a statistical difference, then additional *post-hoc* statistical analysis can be performed to discover where those differences are. Table 3 summarizes the rankings obtained by Friedman procedure. The results highlight single-point as the best operator, so the post-hoc analysis is performed with single-point crossover as the control method.

| Operator | Average ranking |
|---|---|
| Single-point | 3.5667 |
| Two-point | 4.3667 |
| Half-uniform | 4.7667 |
| Uniform | 3.7333 |
| Segmented | 4.4333 |
| Shuffle | 5.4333 |
| Reduced surrogate | 4.3 |
| Non-geometric | 5.9333 |
| Orthogonal array | 9.4 |
| Hybrid Taguchi | 9.0667 |

**Table 3.** Average rankings of crossover operators (Friedman)

With the level of significance $\alpha$ of 0.05 both the Friedman and Iman-Davenport statistic show significant differences in operators with test values of 64.44 and 12.79, respectivelly, and $p < 0.001$.

In the post-hoc analysis we applied the Bonferroni-Dunn, Hochberg, Finner and Li tests [2] over the results of Friedman procedure. The analysis indicates the level of significance with which the control operator is better than each of the remaining operators (i.e. for which the null hypothesis is rejected) and the adjusted $p$ values are shown in Table 4. For the Bonferroni-Dunn test, a *critical difference* (CD) [5] is calculated which for these data equals 3.0656. The interpretation of this measure is that the performance of two algorithms is significantly different only if the corresponding average ranks differ by at least a critical difference, which is depicted in Fig. 1.

It can be perceived that only orthogonal array and hybrid Taguchi operators may be regarded as significantly worse than the single point crossover with a level of significance $\alpha$ of 0.05. For all the other operators, the null hypothesis cannot be rejected with any of the tests for $\alpha$= 0.05. When considering Finner test with level of significance $\alpha$= 0.1, it can be seen that it rejects another hypothesis i.e. single point crossover outperforms the non-geometric crossover.

| Algorithm | unadjusted $p$ | $p_{Bonf}$ | $p_{Hochberg}$ | $p_{Finner}$ | $p_{Li}$ |
|---|---|---|---|---|---|
| Orthogonal array | 0 | 0.000001 | 0.000001 | 0.000001 | 0.000001 |
| Hybrid Taguchi | 0.000001 | 0.000006 | 0.000005 | 0.000003 | 0.000005 |
| Non-geometric | 0.032296 | 0.290662 | 0.22607 | 0.093792 | 0.212294 |
| Shuffle | 0.091322 | 0.821901 | 0.547934 | 0.193838 | 0.432492 |
| Half-uniform | 0.277726 | 2.499538 | 0.880168 | 0.443246 | 0.698581 |
| Segmented | 0.433081 | 3.897733 | 0.880168 | 0.573144 | 0.783272 |
| Two-point | 0.469295 | 4.223652 | 0.880168 | 0.573144 | 0.796594 |
| Reduced surrogate | 0.507122 | 4.564102 | 0.880168 | 0.573144 | 0.808867 |
| Uniform | 0.880168 | 7.921516 | 0.880168 | 0.880168 | 0.880168 |

**Table 4.** Post-hoc comparison (control operator: single-point crossover)

| | SP | TP | HU | U | Seg | Sh | RS | NG | OA | HT |
|---|---|---|---|---|---|---|---|---|---|---|
| Single-point (SP) | 0 | -0.003 | -0.004 | 0.009 | -0.021 | -0.054 | 0 | -0.094 | -3.922 | -3.682 |
| Two-point (TP) | 0.003 | 0 | -0.001 | 0.012 | -0.018 | -0.051 | 0.004 | -0.091 | -3.918 | -3.679 |
| Half-uniform (HU) | 0.004 | 0.001 | 0 | 0.013 | -0.017 | -0.05 | 0.005 | -0.09 | -3.917 | -3.678 |
| Uniform (U) | -0.009 | -0.012 | -0.013 | 0 | -0.03 | -0.063 | -0.009 | -0.103 | -3.931 | -3.691 |
| Segmented (Seg) | 0.021 | 0.018 | 0.017 | 0.03 | 0 | -0.033 | 0.021 | -0.073 | -3.901 | -3.661 |
| Shuffle (Sh) | 0.054 | 0.051 | 0.05 | 0.063 | 0.033 | 0 | 0.055 | -0.04 | -3.867 | -3.628 |
| Reduced surrogate (RS) | -0.001 | -0.004 | -0.005 | 0.009 | -0.021 | -0.055 | 0 | -0.095 | -3.922 | -3.682 |
| Non-geometric (NG) | 0.094 | 0.091 | 0.09 | 0.103 | 0.073 | 0.04 | 0.095 | 0 | -3.827 | -3.588 |
| Orthogonal array (OA) | 3.922 | 3.918 | 3.917 | 3.931 | 3.901 | 3.867 | 3.922 | 3.827 | 0 | 0.239 |
| Hybrid Taguchi (HT) | 3.682 | 3.679 | 3.678 | 3.691 | 3.661 | 3.628 | 3.682 | 3.588 | -0.239 | 0 |

**Table 5.** Contrast estimation (pairwise comparison)

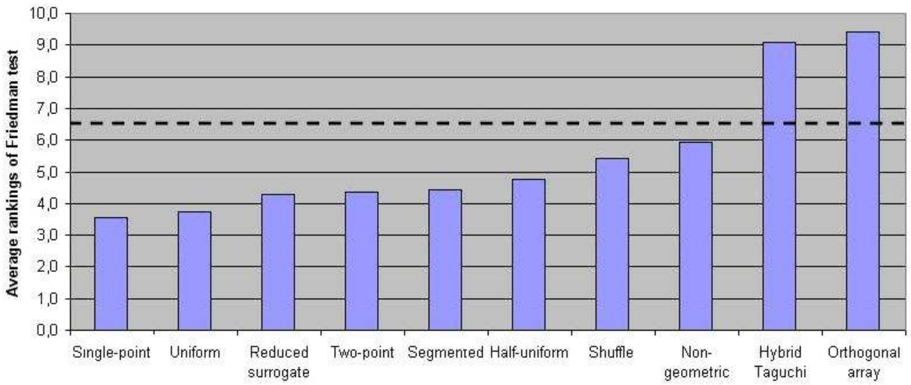

**Fig. 1.** Bonferroni-Dunn's test (CD = 3.0656, control operator: single-point crossover)

Since there are no significant differences for the majority of the operators, a procedure *contrast estimation* based on medians [2] can be used to estimate the differences between each two crossover operators. In this test the performance of the algorithms is reflected by the magnitudes of the differences in error rates, and test values are shown in Table 4. A negative value for the operator in a given row indicates that the operator performs better than the operator in a given column. These results highlight the uniform, single point and reduced surrogate operators as the best performing ones.

## 4 Conclusions

In this work we performed an exhaustive search for optimal choice of crossover operators on a well established set of optimization problems. The results clearly show significant statistical differences between certain operators. On the other hand, the conclusions regarding the best overall operators cannot be given with reasonable significance.

A notable difference can be perceived in the case of orthogonal array and hybrid Taguchi crossover, which perform worse than the other operators on the given set of benchmark functions. Nevertheless, this cannot be taken as a general rule, since there certainly exist some problems for which those operators may behave differently. Among other operators, the most successful ones appear to be the uniform, single-point and reduced surrogate crossover.

The presented results may only be considered relevant if the problem at hand bears similarities with the problems addressed in this work. Since in most cases there is not enough time for an exhaustive operator and parameter search, the provided findings may prove useful to researchers in similar optimization environment.

Further work should consider the following issues: additional experiments with different population sizes and mutation probabilities, tournament selection and additional test functions similar to “real-world” problems. When conducting additional, more exhaustive analysis it is prudent to employ additional statistical tests.

## References

1. Chan, K.Y., Kwong, C.K., Jiang, H., Aydin, M.E., Fogarty, T.C.: A new orthogonal array based crossover, with analysis of gene interactions, for evolutionary algorithms and its application to car door design. Expert Systems with Applications (37), 3853–3862 (2010)
2. Derrac, J., Garcia, S., Molina, D., Herrera, F.: A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Computation (1), 3–18 (2011)
3. Dumitrescu, D., Lazzerini, B., Jain, L.C., Dumitrescu, A.: Evolutionary Computation. CRC Press, Florida, USA (2000)

4. Eshelman, L.J.: The chc adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. In: Foundations of Genetic Algorithms. pp. 265–283. Morgan Kaufmann, San Francisco, CA, USA (1991)
5. Garcia, S., Molina, D., Lozano, M., Herrera, F.: A study on the use of non-parametric tests for analyzing the evolutionary algorithms behaviour: a case study on the cec2005 special session on real parameter optimization. Journal of Heuristics (15), 617–644 (2009)
6. Holland, J.H.: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. The MIT Press, Cambridge, USA (1992)
7. Ishibuchi, H., Tsukamoto, N., Nojima, Y.: Maintaining the diversity of solutions by non-geometric binary crossover: A worst one-max solver competition case study. In: Proceedings of the Genetic and Evolutionary Computation Conference GECCO'08. pp. 1111–1112 (2008)
8. Jakobovic, D., et al.: Evolutionary computation framework. Available from http://gp.zemris.fer.hr/ecf/ (March 2011), http://gp.zemris.fer.hr/ecf/
9. Leung, Y.W., Wang, Y.: An orthogonal genetic algorithm with quantization for global numerical optimization. IEEE Transactions on Evolutionary Computation 5(1), 41–53 (February 2001)
10. Michalewitz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, Berlin Heidelberg New York, USA, third edn. (1996)
11. Mitchell, M.: An Introduction to Genetic Algorithms. The MIT Press, Cambridge, USA (1999)
12. Pohlheim, H.: Geatbx examples examples of objective functions. In: Available as http://www.geatbx.com/download/GEATbx_ObjFunExpl_v38.pdf (2006)
13. Sheskin, D.: Handbook of Parametric and Nonparametric Statistical Procedures. Chapman and Hall/CRC, fourth edn. (2007)
14. Suganthan, P., Hansen, N., Liang, J., Deb, K., Chen, Y., Auger, A., Tiwari, S.: Problem definitions and evaluation criteria for the cec 2005 special session on real parameter optimization. In: Tech. Report, Nanyang Technological University (2005)
15. Tsai, J.T., Liu, T.K., Chou, J.H.: Hybrid taguchi-genetic algorithm for global numerical optimization. IEEE Transactions on Evolutionary Computation 8(4), 365–377 (August 2004)
16. Weise, T.: Global Optimization Algorithms Theory and Application (2009), http://www.it-weise.de/
17. Zhang, Q., Leung, Y.W.: An orthogonal genetic algorithm for multimedia multicast routing. IEEE Transactions on Evolutionary Computation 3(1), 53–62 (April 1999)